

S.B.A.C.E. GAZETTE
South Bay Atari Computer Enthusiasts

VOLUME 1
ISSUE 4
NOV/DEC 1982

The material and opinions in this Gazette are those of the individual authors and do not necessarily reflect the opinions of the South Bay Atari Computer Enthusiasts. The material in this Gazette may be copied by any other User Group providing credit is given to the authors.

Please address all correspondence to:

James A. Jengo
SBACE
5025 Range Horse Lane
Rolling Hills Est., Calif. 90274

As you may already know (or perhaps not) the monthly meetings are held on the 3rd Thursday of each month at 7:00 PM at:

HN Computers
2301 Artesia Blvd.
Redondo Beach, Calif. 90278

Our esteemed officers are:

President	Dan Pinal
Vice President	Jerry Bransford
Secretary/Treasurer	James Jengo
Librarian	Harry Koons
Newsletter Editor	James Jengo
Associate Editor	Sue Whitehead

The SBACE GAZETTE is in no way affiliated with ATARI, Inc. ATARI is a trademark of Atari, Inc.

SBACE
5025 Range Horse Lane
Rolling Hills Est., Calif. 90274



NEW MEMBERS

S.B.A.C.E. welcomes the following new members:

Lloyd A. O'Hara	Al Hunter
Antonio Santa Ana	Allen Davidson
Alan Hicks	Gerald Helping
Robert Beachler	Carl Rice
Gary & Nancy Johnson	Garth Pillsbury
Richard Pearce	Jim Peirce

INNER SBACE (aka Old Business)

Well, believe it or not, we actually got a lot of contributed articles for this issue. Unfortunately (actually, fortunate for me) no one else submitted articles for the last Newsletter besides me, so there wasn't much of a contest for the monthly \$25.00 award for best article. There appear to be some excellent articles this issue so please try to make up your minds as to which one you think is best and we will vote for the best one at the next meeting.

Glorioskees (? spelling)!!! We have gotten a volunteer to be the new Newsletter editor: Sue Whitehead. Sue will take over editorial responsibilities starting with the next issue. THANK YOU Sue.

For those of you who weren't at the last meeting, we heard more good news in the form of a greater member discount at THE store, both for hardware and software ... WOWEE !!! Now there's no excuse for you not to run out and get that fourth disk drive or second line printer or Advent projection TV monitor you've always needed. "Great" you say, "but what am I to do with my old projection TV, etc., when I buy all this new great stuff?" Well it just so happens that I have the answer to your question right here in the form of a new regular column: GOODIES FOR SALE. If you have an old memory board, interface, printer, disk drive, etc., that you no longer want (*poor unwanted thing*), just compose a couple line ad with your name, phone number and the price you want, and place it in the Newsletter box at *HW Computers*, and we will publish it in the next Newsletter.

As mentioned in the last issue, Dan Prince has volunteered to teach a basic Basic class starting 1/2 hour before each regular meeting. Only 5 people said that they would be interested in attending such a class at the last meeting, far fewer than indicated an interest at the prior meeting! If anyone else is interested, let us know at the next meeting, or "forever hold your peace."

As was mentioned at the last meeting, we would *loovve* to give a copy of De Re Atari to the person supplying us with a neato logo for this neato, keeno Newsletter.

OUTER SBACE (What's New)

Another of our members has made the big time: Brenda Bittner (using her maiden name of Brenda Balch) has had a two-part article published in the November and December issues of COMPUTE. These articles involve musical graphics/education. Congratulations Brenda!

Dalton Whitehead has volunteered to edit a regular column on Forth. The first column appears in this issue.

Bill Bacon gave an excellent and very interesting talk on the new Basic compiler from Datasoft which he has been testing and will soon be on the market. This indeed is a very exciting bit (medium pun intended) of software. Bill has written a more extensive description elsewhere in this issue.

Mike O'Shaughnessy and Shawn Rohan (editors of our Game Review column) have done an excellent job and have written several reviews as well as starting a high score column for many popular games.

NEWS FROM OTHER GROUPS

Rumor has it that Epson will be coming out with a new line of printers called the FX series, supposedly faster than the MX ... rumor or fact?

An article on how to modify your 400 computer to 48K/64K (reprinted from the Sept. 1982 M.A.C.E. Newsletter) will be published in the next Newsletter.

ATTENTION AMATEUR RADIO OPERATORS

by Harold Burba N6AXQ

Two Atari microcomputer nets are on every Sunday AM. One is on 20 meters from 9 AM to about 11 AM. The second follows the first and meets on 40 meters for the West Coast members. All Hams with Atari computers or interest in same are invited to check in and meet others with similar interests, problems and/or solutions.

Freq.	Sunday Local Time	Net Control
14.325 MHz	9 - 11 AM	WB8BNG
7.235 MHz	11 - AM	WB601P

Any Hams interested in getting together are invited to call Harold or to let me know if they do not have 'R' under the *Interests* column of the mailing list.

BASIC COMPILER

It's here--a compiler for ATARI Basic is about to be released by DATASOFT. You can use it for your own applications, on commercial products you have bought elsewhere (APX's XREF is a prime candidate), or for software that you develop for sale. In the last case, there will be no run-time licence fee, but you will need to insert a credit referencing the compiler and DATASOFT.

SPEED--this is the objective, and you will find the compiler can increase the speed of your programs by about 15 X.

CODE PROTECTION--since the output is machine code, it is much easier to protect your programs against piracy.

EASE OF DEVELOPMENT AND DEBUGGING--yes, now you can develop your programs in a high-level language, using all the powerful commands, especially in the Graphics area, and do you debugging in interactive mode with BASIC. Then when the program is complete and tested, switch to Machine Code for speed and code protection.

TWO FLAVORS--that's right, you have the choice of selecting the floating point option at compile time, and getting around 3 X the speed, or the 2-byte integer option and getting the full 15 X.

An update that is expected to be released later will allow 4-byte integer for around a 5 X speed increase. The FP option is most suited to scientific and mathematical applications. The 2-byte integer option is most suited to utilities and games. The 4-byte integer version will be the preferred option for business applications, but until it comes out, use floating point.

LARGEST PROGRAM--the largest program that you can compile will be around 150 sectors.

DOUBLE DENSITY DRIVES--yes, it works fine with the PERCOM drives. In fact, the largest program compilable increases to around 160 standard sectors (80 DD sectors) with these drives.

MEMORY--you will need 48KB. A compiled program will be larger than the BASIC version--ballpark is about 20% larger. This is so of most assembler programs, since BASIC is actually a form of shorthand, which is expanded, one line at a time, when the program is run (this is also why BASIC is so slow).

FORMAT--the compiler takes your program, in tokenized mode, and generates assembler code from it in PASS 1. PASS 2 builds a symbol table from the generated assembler code. PASS 3 assembles the code from the compiler and the appropriate run-time library into an executable machine language program. The Run-time library is assembled beginning at \$2400 (decimal 9000). The program itself is assembled beginning at \$3200 (decimal 12500). PASS 4 gives you the option of listing to the screen or the printer a BASIC statement #/memory address cross-reference. This is very useful, since if you get a system error in program execution, you can abort the run, continue from the point of error, restart your program at the beginning, or start it at any other point (aren't you glad you took a hard-copy of the addresses?).

You can also immediately execute the program you just compiled while in PASS 4.

RESTRICTIONS--I know this question was high in your mind. The news is excellent. ALL OF ATARI BASIC IS IMPLEMENTED, with the exception of commands that deal with program files (SAVE, CSAVE, LOAD, CLOAD, LIST, ENTER, RUN with an argument--RUN by itself is OK), and the commands DOS (yeah, I wish that one was there too!), NEW, BYE, and CONT. None of that is too hard to live with (well, DOS would have been nice, and without arguments after the RUN command, you can't chain programs). You can only terminate a FOR/NEXT loop with a single NEXT (no NEXT's in IF statements--use GOTO's to the NEXT), you can't use variable line #'s (but you can use ON GOTO/GOSUB to get the same effect), and you can't PRINT CHR\$(155) (use PUT #n,155 instead).

You must DIMension arrays in your program before any line that references them--not just before in the logical flow of the program, but in a line with a lower line #!!! Also, all DATA statements must be at the end of the program. There is also a restriction on the way strings and substrings are handled that means you will avoid having substrings on the left side of an "=" sign, or assigning to any string variable a literal that contains a character with the decimal value 155 (since this is the RETURN code in BASIC, you would have had to POKE this into the literal in the first place, so do it to the string instead). The restriction on substrings on the left side of an "=" is handled with PEEK's and POKE's, so you can still do what you need to do in your program, just do it in a different way.

INTEGER OPTION--you can't use the transcendental functions (SIN,COS,ATAN,CLOG,LOG,EXP,SQR), of course, or any fractions, since the only numbers supported are integers. And, being 2-byte integers, the numbers range from -32767 to 32767 (you can, however, PEEK and POKE up to 64K and the compiler will generate workable code for you). When using RND, instead of saying 4*RND(0) to get a number between 0 and 3.9999999, you say RND(4) to get an INTEGER between 0 and 3. To use the smallest amount of memory possible, use GOSUB's liberally (isn't it wonderful to be able once again to follow the precepts of Structured Programming, and Top-down systems design & programming & testing--much better programs should result, with fewer bugs, clearer documentation and more reliable maintenance), while minimizing the use of SUBSTRING's, ARRAY's or a variable as the step size in a FOR/NEXT loop.

Since your programs will now run faster than excrement through a goose, you will have to stop using FOR/NEXT loops to build delays into your program (you shouldn't be using them anyhow, since they are dependent on position in your program)--instead use:

```

.....
nnnnn D=60:GOSUB 20000
.....
20000 POKE 544,D
20010 IF PEEK(544)>0 THEN 20010
20020 RETURN

```

to give delays measured in "jiffies", or 1/60 of a second. The preceding example gives a one second delay. Since you can POKE any number up to 255, you can have any delay up to just over 4 seconds. If this is not enough for you, use the following:

```

.....
nnnnn D1=88:D2=2:gosub 20000
.....
20000 POKE 544,D1:POKE 545,D2
20010 IF (PEEK(544)+PEEK(545))>0 THEN 20010
20020 RETURN

```

to give delays of up to just over 18 minutes. D1 is # of jiffies and D2 is # of 256 X jiffies. The preceding example gives a delay of 10 seconds.

MY COMMENTS: This product is HOT!!! Look for ATARI to explore this as a program product in the near future.

(P.S. While you can run this product on a single density, single drive system, it is easier with two single density drives, and easier still with one double density drive. H & W has a good price on PERCOM double density drives, and I can speak from experience that they are great--I have two of them)

by Mike O'Shaughnessy
and Shawn Rohan

ZORK I

Shawn and I have been following the Interlogic adventure series. As soon as Zork I was produced by Infocom, we purchased it. Zork I is simply incredible, with all the individual puzzles to solve in almost every room. At times we were completely stumped, but at other times we went through the different rooms with no problems at all.

As I said, Zork I is available from Infocom Inc. To operate Zork I, you need the following: Atari 400/800, 32k RAM, and one disk drive. Optional items are: one or more formatted diskettes for saving game positions, 40/48k RAM for faster execution, and a printer along with an 850 Interface for scripting.

Zork I is an adventure game in which you explore different rooms, forests, canyons, why you can even sail in a magical boat! Your main objective is to find 20 treasures that are littered about the adventure.

Zork I is capable of understanding whole phrases and sentences, and has a 600+ vocabulary.

An InvisiClues booklet is available for Zork I and Zork II. These clues are produced by the Zork Users Group. Each booklet contains over 175 different hints and answers to over 75 different questions. The clues are printed in invisible ink (developing marker included), giving you the option to develop only what you want to see. The Clues are available by mail-order for \$9.95.

Shawn and I enjoyed this game thoroughly. We gave Zork I a 7.5 on our game scale. It is highly recommended for people who like games that involve problem solving. Read the newsletter next month when we will feature Choplifter and Clowns & Balloons.

Game Review Centipede
by Anthony L. Sena

Too much flight time have you weary? Had your fill of dots? Take a trip through a "Magical" mushroom patch with Atari's latest release - Centipede.

It's Saturday morning and like a typical middle class homeowner you're out in the yard looking at the weeds wondering where to start. You bend to pull a mushroom from the damp grass. Ouch! A centipede has bitten you. A bug sprayer is close by and you grab it, but before you can shoot a strange dizziness overcomes you and you begin to shrink! The plants loom over you and the mushrooms that you were standing by are now giving you shade as you stare in shock at your distorted reflection in a drop of dew. A loud drumming sound brings you out of your daze. Across the lawn, marching through the mushrooms, you see it. A huge centipede! In your hand is the sprayer and you begin to shoot wildly. With each body section you destroy, the bug splits into two shorter centipedes but keeps coming! Before the centipede gets close enough to bite, a spider jumps at you from your blind side. Luckily you've had your morning coffee and are alert enough to drop him in his tracks.

Centipede has other "beasts" to plague you including a frenzied flea (worth 200 points dead), and a scorpion (1000 points) who poisons the mushrooms to drive the centipede mad. A 900 point reward is given for steel nerves and a quick trigger finger if the spider is blasted at extremely close range but stunts like that are better left to expert bug blasters.

As usual for Atari cartridges the space bar pauses the game for those who need a break. Two can play (one at a time) and high score is displayed until somebody turns off the game (I wish Atari had bubble memory when jealous friends are around). Lord Motley Bugnut gives a nice talk on bug blaster strategy in the booklet. 16K RAM required.

GOOD SHOOTING!

PS. To find out how the story ends just run over to HW Computers and pick up a Centipede.

The following is an article taken off of Jonith Johnson's BBS . In order to get on the BBS, hook up a modem set to 300 baud to your computer and call (213)638-3204.

Sue Whitehead

Bulletin Board

In 1978, Ward Chistiansen and Randy Suess designed and implimented the first computerize bulletin-board system. The ideal was to let local computer club members call in and leave a message, check for messages sent to them and transfer their favorite programs to or from the system. We carry on the same tradition with the Atari BBS.

People have an overwhelming need to communicate with one another. Technology reflects the need; a large part of technology progress has been geared toward improving communication methods and satifying this urge.

Because this buttetin-board is base on the Atari computer and is called by all types of computers systems, we used the standard ASCII communication mode. It's a good idea to get into the special Atari mode, called ATASCII, if you have an Atari computer, this way you can recieve all of the inverted video and the special Atari characters that are unique to only the Atari computer . Some games will not work properly unless you download them in this mode. To see how to get in this mode, read the user's manual on drive #1 (Hit the ' M ' key) on this system.

To get on line with this board simply dial the phone number 213-638-3204 you'll hear a carrier signal(a high-pitch tone)after the phone rings a couple of times. When you get a carrier signal, place the headset in the modem cradle (for a acoustic modem) or turn on the modem for direct connect.

This system requires that you hit the < RETURN > two times to signal the board of your presence. The board will then tell you a connection has been established and give you the time and date. The system then ask for your name, city, state, which will be kept on disk for statistical purposes.

This board has a command menu and an expert mode to used if you're already familiar with the commands. You can toggle between expert and normal mode with the ' X ' key. Any time you need help with the commands hit ' H ' for help.

To see the files on the drives use the ' F ' command for files.

I hope you enjoy your stay on this system. Feel free to leave messages on the board at any time, also leave any comment on the printer at log-off time. I assure you all comments are read and taken note of.

Happy computing !
SYSOP: Jonith Johnson

" FORTH & BACK "

By D.J. Whitehead

This is an article for the newcomer to FORTH. I am assuming you have no prior experience with the FORTH environment. I am also assuming, that you are short on \$\$\$, otherwise you could have purchased a FORTH software package which would have included how to use it.

In order to gain some experience with FORTH, we will use the public domain version in the SBACE library. So much for the good news, now for the bad... you must have an ATARI 810 disk drive (or equivalent) to be able to use the library copy.

To copy the FORTH disk using the FORTH copy routine you will need an ATARI with 40k or more with cartridges removed and two disk drives. In a future article I will tell you how you can edit the copy routine to require only one drive. As to the minimum amount of memory required, that will vary based on how complex you program your FORTH environment.

COPYING FORTH

Required:

- ◇ 1 ATARI with 40k minimum
- ◇ 2 810 disk drives (or equivalent)

PROCEDURE

- 1) Remove all cartridges
- 2) Turn on both disk units
- 3) Wait until the busy lights on the drives are off
- 4) Insert the public domain FORTH into drive number 1
- 5) Insert a FORMATED disk into drive number 2.
- 6) Turn on the ATARI 800 unit.

After a few seconds, the FORTH banner will come up on the screen. If this does not happen, and the drives have stopped, then you have a problem. In this event you better recheck your set up. If all is OK, then leave a note in the library, addressed to me, saying what you did, and what machine you used. (It is possible to damage the FORTH disk, if this happens I will replace it back for the SBACE users.)

Once in the FORTH system, the following commands must be executed to generate your copy.

- 1) Type 36 LOAD then <RETURN>
(Note the space between the 36 and LOAD)
- 2) Once the loading process has been completed (the system prints "OK") you are ready to copy the disk.
- 3) Check that a formatted disk is in drive # 2.
- 4) Type DISKCOPY then <RETURN>
- 5) At this time drive # 1 should start, followed by drive # 2.
- 6) On the screen "90" should appear, showing 90 sectors have been copied.
- 7) Items 5 and 6 should be repeated until the FORTH disk has been copied. Note the screen will show "90 180 270 ..." etc, counting off the total number of sectors copied to drive #2.
- 8) You are finished, when "OK" appears on the screen.

CHECK YOUR COPY!!!

This can be done by waiting until both drives have stopped. Then take the public library disk out of drive #1, and replace it back into the SBACE Library. Take your copy out of drive # 2 and place it into drive #1. Turn off the ATARI 800, and then turn it back on. If all is well, you will be greeted with the FORTH banner.

If you are not greeted with the banner, then back to square 1 and try again.

Now you have YOUR copy of FORTH ! Here are a couple of things you can try.

Under the FORTH vocabulary (more about vocabularies later) you can type VLIST then <RETURN>. VLIST will print to your screen a table of all the valid commands (FORTH words).

FORTH has vocabularies in which definitions of words can be located. So far we have only used one vocabulary called the FORTH vocabulary. However the disk which you have copied has two additional vocabularies: EDITOR and ASSEMBLER. For more information about the ASSEMBLER vocabulary I suggest any good book on 6502 assembler code, along with STARTING FORTH by Leo Brodie as references.

The vocabulary called EDITOR will prove useful (how else can you program?). In order to gain entrance to the EDITOR, type EDITOR and <RETURN>

Once in EDITOR you can again type VLIST in order to determine the EDITOR commands (FORTH words)

A few useful EDITOR words are:

◇ LIST... as in 36 LIST

(This lists screen 36 to the edit buffer, and the screen)

◇ L... used to list whatever is in the edit buffer to the screen.

◇ xx P new text.... <RETURN>

where xx is the line number from 0 to 15

(This is used to over write line numbered xx on the current edit buffer.)

◇ FLUSH... used to flush all updated buffers to the disk.

WARNING --> FLUSH will delete the old disk screen, replacing it with whatever you have in the ATARI buffers (even trash.)

This will give you a copy of FORTH. I plan to write more articles on FORTH for future issues of the newsletter. Some topics will be how to modify DISKCOPY to allow single disk copy and give a graphics demo to draw to the screen with a joystick.

GETTING TO KNOW (AND LOVE) ASSEMBLY LANGUAGE

by James A. Jengo

Welcome back! Ready to try some more? This time we'll continue with addition and learn to subtract, both using 2-byte numbers (the way you usually have to do it).

Remember, the high byte contains the number of "256's", and the low byte contains the "remainder" that is added to the product of the value stored in the high byte 'times' 256. Therefore if we add two low bytes together and the new sum is greater than 255 then we INCRement the high byte counter by 1, and zero out the low byte. Conversely, when subtracting if the difference between the low bytes is less than 0, then we simply DECrement the high byte of the answer and take the '256' that we "borrowed" from the high byte and add it to the negative difference value, thus ending up with a valid (between 0 and 255) value.

That all sounds very good (who am I kidding) but, you ask, how do I (or actually the computer) know when the value of the A register (where addition and subtraction operations are performed) is < 0 or > 255 ? Well it just so happens that there is a flag called the Carry flag that is SET if the A register value exceeds 255 and is CLEARED if the A register value becomes less than zero. Therefore all we have to do is to check to see if the Flag is changed by an operation and, if so, then either DECrement or INCRement the high byte of the answer, as appropriate. Hopefully the following example will help clarify this:

```
10 ;2-byte Addition Example
20 *= $600 ;Tell computer that this program will reside in memory at
    starting location $600 (1536)
30 ;
40 ;Set aside 6 bytes of memory, each to represent the high or low
    byte of a particular variable that will be passed over from the Basic
    program, or the answer (SUM)
50 ;
60 FRSTHI .BYTE 0 ;assign an arbitrary value of '0' just to reserve
    this space
70 FRSTLO .BYTE 0
80 SCNDHI .BYTE 0
90 SCNDLO .BYTE 0
100 SUMHI .BYTE 0
110 SUMLO .BYTE 0
120 ;
130 ; Pull the high and low byte values for the First and Second
    numbers from the Basic program
140 ;
150 PLA ;pull the # of variables to be passed ... and forget it (who
    cares)
160 PLA ;pull the high byte of the first variable passed from Basic
170 STA FRSTHI ;store the high byte (which is in the A register) as
    the variable named FRSTHI
180 PLA ;pull the low byte of the first variable passed from Basic
    and place it in the A register
190 STA FRSTLO ; and store it in the variable named FRSTLO
200 PLA
210 STA SCNDHI ; do the same thing for the second variable
220 PLA
230 STA SCNDLO
240 ;
250 ; Now do the addition
260 ;
```



```

270 LDA FRSTHI ; place the high byte of the first number in the A reg
280 CLC ; Clear the Carry flag so we can later check if it is
           still clear or if it got Set as a result of the
           operation
290 ADC SCNDHI ; ADd with Carry flag effected, the high byte of the
           second number
300 STA SUMHI ; store the resultant high byte sum (in the A register
           ) in the variable named SUMHI
310 LDA FRSTLO ; do same thing for low byte...
320 CLC
330 ADC SCNDLO
340 BCC NEXT ; Branch if the Carry flag is still Clear (i.e. the sum
           in the A register is not > 255, to the line labelled
           as 'NEXT'
350 INC SUMHI ; if the Carry flag was not clear at the time of the
           last instruction, this line will be executed, thus
           indicating that the A register content exceeded 255.
           Therefore the high byte counter (SUMHI) is INCRement
           -ed by 1
360 NEXT STA SUMLO ; store the sum of the low bytes in the variable
           SUMLO
370 ;
380 ; Pass the sum of the two numbers (stored as SUMHI and SUMLO back
           to the Basic program
390 ;
400 LDA SUMHI
410 STA $D5 ; store the high byte of the sum into memory location $D5
           (the location where the returning high byte for a USR
           call from Basic must be stored)
420 LDA SUMLO
430 STA $D4 ; same thing for the low byte
440 ;
450 RTS ; Return To Subroutine command (returns control back to the
           Basic program)
460 .END ; tells the assembler that this program is complete

```

This assembly language program can be called from Basic with the USR command as demonstrated by the following program:

```

10 PRINT "ENTER FIRST NUMBER";:INPUT FIRSTNUM
20 PRINT "ENTER SECOND NUMBER";:INPUT SECNDNUM
30 SUM = USR (1536,FIRSTNUM,SECNDNUM)
40 PRINT "SUM = ";SUM
50 END

```

The next assembly language program is analagous to the first one but represents 2-byte subtraction:

```

10 ; 2-byte Subtraction (assumes that the first number is > second #)
20 ;
30 *= $600
40 FRSTHI .BYTE 0
50 FRSTLO .BYTE 0
60 SCNDHI .BYTE 0
70 SCNDLO .BYTE 0
80 DIFFHI .BYTE 0
90 DIFFLO .BYTE 0

```

```

100 LDA FRSTHI
110 SEC ; SET the Carry flag
120 SBC SCNDHI ;subtract and leave the Carry flag set if the result
      in the A register is not < 0, otherwise Clear the
      Carry flag
130 STA DIFFHI
140 LDA FRSTLO
150 SEC
160 SBC SCNDLO
170 BCS NEXT ; Branch if Carry flag is Set to the line 'NEXT'
180 DEC DIFFHI
190 NEXT STA DIFFLO
200 ;
210 LDA DIFFHI
220 STA $D5
230 LDA DIFFLO
240 STA $D4
250 RTS
260 .END

```

As mentioned in the last issue, the code could be streamlined quite a bit, but this way the logic is easier to follow.

Well that's about it for now. See you next issue.

DISKETTE DIRECTORY POCKETS by Mike Markowitz

Have you ever wondered what to do with those neat directory lists that DOS allows you to make. Well, I think you will like this idea!

First of all, to make yourself a print out of the disk directory, you bring up DOS to the screen and select item "A" and press RETURN. Then type "D2:,P:" when prompted for the "DIRECTORY-SEARCH SPEC, LIST FILE?". This procedure will list all files on the diskette in DRIVE #2 to the printer.

To make the jacket pocket you must now go to a good office stationary store and purchase packing list label sleeves. One such type is "Dennison, U.S.A. # 17-215" which costs approximately 10 cents each. They measure 5 1/4 x 4 1/8, and have a clear window with "PACKING LIST ENCLOSED" printed on a red background across the top of the sleeve. To use this and apply it to the bottom front of your diskette cover, you must first cut off the top of the sleeve just under the printed part, leaving the bottom with only the clear window. It will take a little practice to properly line it up while sticking the sleeve to the diskette jacket. I would suggest you waste one or two sleeves on a piece of paper first to get the hang of it. Once attached to the jacket you can easily insert your new directory in the sleeve, and change it whenever necessary. I have found this to make it very easy to keep track of what's on each disk. I hope you will too. ENJOY !!!

Well, that's it for this issue. Remember that we do need YOU to contribute articles, and that if you do you will qualify for the \$25.00 prize and the extra store discount.
HAPPY HOLIDAYS !!!